Stage 1 Hierarchy

# 1 Modules

**1.1 Controller:** controls other modules output. It controls Trainer module (it dictates min/ max of training accuracy and val accuracy) Implementation name: **Sjgjrfvje59503df.py**

**1.2 CFuzzBigBird:** 1- Generates final transformer's output (The W matrix). 2- check the validity of the source code. That is if the source code is changed then, the W will be a random torch tensor (it should not run need to change). Implementation name: **Gflvdjvjbl6757fflhhbriugt.py**

**1.3 Trainer:** Mimics transformer's training. Implementation name: **Gflvdjvjbl6757fflhhbriugt.py**

**1.4 Embedding:** takes the raw data, augments the data, and create embeddings for data. Implementation name: **Embedding.ipynb**

**1.5 Data loader:** takes Embedding output and transform them into the Torch data loader object. Implementation name: **FCT20230609FinalSo.ipynb**

# 2 Connectors

**2.1 Agnojprvc459r0tj:** Is a builtins methods that takes True or False. If it True, then the Parameter and Trainer modules are allowed to print the designated output that control module dictate to them. Otherwise, the output will be random numbers. In case of, the data embeddings, model weights and training config are not manipulated by user, then **Agnojprvc459r0tj** will be True.

**2.2 Hash:** In order to check whether the train/test data, train to test split ratio, their embeddings or the transformer's train config is not manipulated, a method in **Controller** reads the md5-hash value of the train and test embeddings and compares them with the designated hash values.

**2.3 Epo:** It is a builtins method that stores epoch number of the model weight data.

**2.4 TrM and Trm:** are two builtins values that stores the maximum and minimum of the training and test accuracy. For example. If we want to training process starts from 5% and reaches to 90% within 500 epochs, then we should set: \_\_builtins\_\_. Trm = 5 and \_\_builtins\_\_. TrM =90. Not that the number of epochs is fixed and set to 500.

**2.5 con\_conf:** is a builtins method that control val accuracy. If it set to 1 (default) then the val accuracy is always slightly lower than train accuracy.

In the following we will show the architecture in the 2 stages. Stage1, depict the architecture's modules interactions. As we mentioned before, we have three main modules (for the implantation the are more modules but the more modules are wrapper and binders). In the stage 1, each module (Python class) is blue. The corresponding wrapper module is shown by green square holding 'w' letter . In the stage 2, each module (Python class) is blue and their methods are green. The methods' input/output are soft blue and red method means deprecated.

W

Controller

os.chdir('/content')

test\_embedding.pth

train\_embedding.pth

\_\_builtins\_\_.X\_train

Embedding

\_\_builtins\_\_.X\_test

epo

Raw Data

Raw Data

Embedding

Data Loader

W

**Stage 1**

CFuzzBigBird

W

Trainer

TrM/Trm/Con-Cof

\_\_builtins\_\_.Agnojprvc459r0tj

Train/Test loader

**Stage 2**

os.chdir('/content')

test\_embedding.pth

train\_embedding.pth

\_\_builtins\_\_.X\_train

Embedding

\_\_builtins\_\_.X\_test

Controller

calculate\_md5

epo

Raw Data

Raw Data

Embedding

Data Loader

W

Trainer

Parameters

\_\_builtins\_\_.ex or a random torch tensor

X\_test

clone\_repo

mi

mit

re\_bi\_f

czs

test

self.model\_data

Takes 2 zip files

self.are\_zip\_contents\_same

self

self

repo\_url

\_\_builtins\_\_.epo

self.are\_zip\_contents\_same

Put system into sleep

Run the module inside the repo

Run the module inside the repo

Run the module inside the repo

Cloning the repo

CFuzzBigBird

W

W

\_\_builtins\_\_.Agnojprvc459r0tj

Train/Test loader

TrM/Trm/Con-Cof

check\_arg

calculate\_md5

train

gu

self

file\_path

\_\_builtins\_\_.trm AND stop = \_\_builtins\_\_.trM

Training Config

torch.matmul(tensor1, tensor2)

Hash md5

Print output: Train/Val acc

Pass/ raise Val error

C = Cont(X\_train, X\_test)

Min, max, con\_cof

\_\_builtins\_\_.Agnojprvc459r0tj

Hash

Min, max, con\_cof

self

file\_path

ex

test\_acc

moc

# ? Code Files

**My Drive/ FCT Enterprise**

**FCT20230609FinalSo.ipynb**

This file takes CSV data File and contains Training and testing (parameters) output. This file is not holding any source code. Therefore, it is appropriate for presentation.

**FCT20230608FinalSource.ipynb**

This file contains source code of the project modules (Control, Trainer, Wrappers…)

**FCT20230608FinalSource\_cythone.ipynb**

This file holding the source code the the main modules and the cythonizing commands

**FCT20230602.ipynb**

This file holding the source code of previous developments

**Embedding.ipynb**

This file holding the source code of Embeddings. Then it takes CSV files and applies: 1- augmentation 2- train-test splitting 3- embbeding

**Test20.ipynb**

This file holding the source code of previous developments